Omtentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap
CTH HT-15, TDA540

Dag: 2016-08-25, *Tid:* 14.00-18.00

Ansvarig:	Alex Gerdes
Examinator:	Joachim von Hacht och Christer Carlsson
Förfrågningar:	Alex Gerdes (alexg@chalmers.se, 0317726154)
Resultat:	Erhålls via Ladok
Betygsgränser:	3:a 24 poäng 4:a 36 poäng 5:a 48 poäng max 60 poäng
Siffror inom parentes:	Anger maximal poäng på uppgiften
Granskning:	Tentamen kan granskas på den 1:e och 8:e september 2016, 12:00 – 13:00 i EDIT 6128. Vi eventuella åsikter om rättningen ange noggrant vad du anser är fel.
Hjälpmedel:	Cay Horstmann: *Java for everyone* eller Jan Skansholm: *Java direkt med Swing* Understrykningar och smärre förtydligande noteringar får finnas.
Var vänlig och:	Skriv tydligt och disponera papperet på lämpligt sätt. Börja varje uppgift på nytt blad. Skriv ej på baksidan av papperet.
Observera:	Uppgifterna är ej ordnade efter svårighetsgrad. Titta därför igenom hela tentamen innan du börjar skriva. Alla program skall vara väl strukturerade, lätta att överskåda samt enkla att förstå. Indentera programkoden! Vid rättning av uppgifter där programkod ingår bedöms principiella fel allvarligare än smärre språkfel.

Lycka till!

Uppgift 1

(3 poäng)

Vad blir utskriften när nedanstående program exekveras?

```
public class Swap {
  public static void main(String[] args) {
    int i = 42;
    int j = 1337;
    System.out.println("x = " + i + " och y = " + j);
    swap(i, j);
    System.out.println("x = " + i + " och y = " + j);
  }

  public static void swap(int i, int j) {
    int temp = i;
    i = j;
    j = temp;
  }
}
```

Uppgift 2

(3 poäng)

Vilken utskrift fås då nedanstående program körs?

```
public class SwapArr {
  public static void main(String[] args) {
    int[] xs = {1, 2, 3, 4, 5};
    print(xs);
    swaparr(2, 4, xs);
    print(xs);
  }

  public static void swaparr(int i, int j, int[] xs) {
    int temp = xs[i];
    xs[i] = xs[j];
    xs[j] = temp;
  }

  public static void print(int[] xs) {
    for (int x : xs) System.out.print(x + " ");
    System.out.println();
  }
}
```

Uppgift 3

(2 poäng)

Vad skrivs ut när följande kodsegment exekveras?

```
int n = 123;
int m = 0;
while (n != 0) {
  m = m * 10 + n % 10;
  n /= 10;
}
System.out.println(m);
```

Uppgift 4

(3 poäng)

Nedanstående kodavsnitt har några (kompilerings)fel. Förklara vad som är fel och rätta till.

```
public static int power(int x, int n) {
  int m = 1
  for (int i = 0; i < n; i++)
    m *= x;
}
```

Uppgift 5

(7 poäng)

En digital bild kan representeras som ett tvådimensionellt fält av bildpunkter. I en digital färgbild utgörs varje bildpunkt av tre heltalsvärden i intervallet 0-255, där de enskilda värdena representerar intensiteten av färgerna rött, grönt och blått. En färgbild kan avbildas med ett tredimensionellt fält av typen `int[][][]`. Den första dimensionen definierar bildens höjd, den andra dimensionen definierar bildens bredd och den tredje dimensionen representerar färgerna rött, grönt och blått. Din uppgift är att skriva en metod

```
public static int[][][] cut(int[][][] samples, int x, int y, int n, int m)
```

som tar en bild (`samples`) och skär ut en del av bilden som blir returnat som en ny bild. Delen som ska skäras ut börjar på punkt `(x, y)` och är `n` gånger `m` pixlar stor. Till exempel:

(a) Original

(b) Del

Man får anta att indata är korrekta.

Uppgift 6

(8 poäng)

Det månatliga återbetalningsbeloppet för ett lån beräknas med formel:

$$belopp = \frac{lc(1+c)^n}{(1+c)^n - 1}$$

där l är lånebeloppet, c är månadsräntan (en tolftedel av årsräntan) och n är antalet månader för att avbetala lånet.

Din uppgift är att skriva ett program som upprepade gånger läser in lånebelopp (i kronor), årsränta (i procent) och återbetalningstid (i år), samt beräknar och skriva ut det månatliga återbetalningsloppet. Du får själv välja om du vill göra in- och utmatning via dialogrutor eller använda `System.in` respektive `System.out`.

För att erhålla full poäng på uppgiften:

- skall programmet utformas på så sätt att inläsningen upprepas tills användaren avbryter exekveringen (vid användning av dialogrutor genom att användaren trycker på Cancel-knappen och vid användning av `System.in` genom att användaren lämpligen ger ctrl-z eller ctrl-c),
- skall lånebelopp, ränta och återbetalningstid läsas i en inläsningssats (dvs ett Scanner-objekt skall användas)
- skall en felutskrift ges om någon av indatan som läses in inte är positiv
- skall beloppet i utskriften anges som ett heltal
- skall programmet innehålla en metod

  ```
  public static double calculatePayment(double loan, double rate, int years)
  ```

  som beräknar det månatliga återbetalningsbeloppet

Uppgift 7

(8 poäng)

För att ett svenskt personnummer skall vara giltigt måste det gälla att:

- det är 10 siffror långt,
- varje siffra är mellan 0 och 9,
- kontrolsiffran är giltig (se nedan).

Kontrollsiffran beräknas på följande sätt:

1. Multiplicera de 9 första siffrorna med omväxlande 2 och 1
2. Produkterna ska sedan splittras upp och adderas
3. Denna summa subtraheras från närmast högre tiotal, eller från sig självt om den är jämnt delbar med 10, detta kan man beräkna så här: $10 - (summan\%10)$

Exempel

Anta att personnumret är 8112189875. Beräkningen av kontrolsiffran tillgår på följande sätt:

1.

siffra	8	1	1	2	1	8	9	8	7
omväxlande 2 och 1	2	1	2	1	2	1	2	1	2
produkterna	16	1	2	2	2	8	18	8	14

2. Summan blir: $1 + 6 + 1 + 2 + 2 + 2 + 8 + 1 + 8 + 8 + 1 + 4 = 44$
3. Kontrollsiffran blir: $10 - (44\%10) = 6$, vilket innebär att personnumret är ogiltigt

Din uppgift är att skriva en metod:

```
public static boolean checkPersonNumber(String number)
```

som tar ett personnummer som argument och avgör om det är ett giltigt personnummer eller inte.

Uppgift 8

(9 poäng)

Skurlängdskodning (engelska: Run Length Encoding) är en datakompressionsmetod som är vanlig då man har data där samma skrivtecken ofta uppträder flera gånger i följd. En sekvens med upprepningar av samma tecken brukar kallas skurlängd. I skurlängdskodning utförs komprimeringen genom att finna skurlängder av tecken och ersätta dessa med enbart ett enkelt tecken och en siffra som antyder hur många gången tecknet upprepas, vilket resulterar i kortare längder för att uppnå komprimering. Till exempel:

```
"JJJJJJJJJJAAAAAAAAAAAAAAAAVVAAAAAAAAAAA"  =>  "11J16A2V11A"
```

Skapa en klass `RLE` med följande metoder:

1. `public static String compress(String input)` som returnerar en komprimerad sträng enligt skurlängdskodningmetoden; anta att `input` strängen innehåller enbart bokstaver
2. `public static String decompress(String input)` som tar en komprimerad sträng som argument och returnerar dekomprimerade strängen

För att lösa uppgifterna är det tillåtet att använda metoder från klassen `String`, till exempel:

- `char charAt(int i)` ger tecknet vid index `i`
- `int indexOf(char ch)` ger index för tecknet `ch` eller -1 om tecknet saknas
- `int length` ger längden av strängen
- `String subString(int start, int end)` ger en delsträng från `start` till `end - 1`
- `String subString(int start)` ger en delsträng från `start` till strängens slut
- `String[] split(String str)` delar upp en sträng i ett fält av delsträngar utifrån ett visst tecken, till exempel:
  `"aaa:bb:cccc:dd".split(":") -> {"aaa", "bb", "cccc", "dd"}`

Uppgift 9

(8 poäng)

1. Skriv en klass `Point` som implementerar följande gränssnitt:

   ```
   public interface Coordinate {
     public void setX(int x);
     public int  getX();
     public void setY(int y);
     public int  getY();
   }
   ```

   Se till att man kan skapa ett objekt, om `x` och `y` har typen `int`, så här:

   ```
   Point p = new Point(x, y);
   ```

   Objektet `p` har typen `Point` och värdet av `p.getX()` är lika med `x` och `p.getY()` är lika med `y`.

2. Förklara skillnaden mellan:

   ```
   Point p = new Point(1, 2);
   ```

   och

   ```
   Coordinate p = new Point(1, 2);
   ```

3. Skapa en subklass av `Point` som heter `Pixel` som inte bara implementerar gränssnittet `Coordinate` men också följande gränssnittet:

   ```
   public interface Color {
     public void setColor(int color);
     public int  getColor();
   }
   ```

Uppgift 10

(9 poäng)

Conway's Game of Life, är en cellulär automat där användaren väljer en startuppställning som helt avgör hur simulationen kommer att utvecklas, enligt strikta regler. Utmaningen ligger i att hitta startuppställningar som ger intressanta utvecklingar. Game of Life är ett exempel på hur komplicerade mönster kan uppstå från väldigt enkla regler.

Simuleringen sker på ett tvådimensionellt rutnät. Rutorna (cellerna) kan vara på eller av. Brädets utseende förändras enligt följande regler:

1. En cell föds om den har exakt tre grannar. Som grannar räknas direkt intill-liggande rutor horisontellt, lodrätt eller diagonalt
2. En cell dör om den har färre än två grannar (isolering) eller om den har fler än tre grannar (trängsel)

I övrigt förblir cellen oförändrad. Huruvida en cell skall förändras skall beräknas innan någon cell förändras, man måste med andra ord räkna ut hela brädet innan man går över till nästa tur (generation).

1. Ange 3 lämpliga klasser
2. Hitta något (eller några) lämpligt attribut (instansvariabel) för varje klass
3. Hitta någon (eller några) lämplig metod för varje klass